Assignment 1: Strings & Classes

Welcome to your first assignment! Now that you have learned the basics of classes and methods as well as how to use variables to store data of various types, you're ready to put these skills into action!

The activities in Week 1 Recitation will help you practice for this assignment, so it is highly recommended that you attend the live session or review the recordings and be sure to attempt the recitation activities.

Learning Goals

This assignment is designed to reinforce the following concepts:

- Using an in-built class in Java
- Navigating the official documentation that Oracle provides you
- Translating a basic algorithm (sequence of steps) into real code
- Creating a class according to specifications

Questions

Question 1. We have seen how to use the String class in Java. One big constraint with String is that the object cannot be modified after it has been created. (Note that methods like `toUpperCase` and `replace` return a new String object and don't change the original object.) For this question, you will explore the StringBuilder class in Java that lets you modify existing objects. Start by looking through Oracle's tutorial on StringBuilder:

https://docs.oracle.com/javase/tutorial/java/data/buffers.html

and the Javadocs on StringBuilder:

https://docs.oracle.com/javase/8/docs/api/java/lang/StringBuilder.html

Once you've done this, write a Java program called **StringBuilderExperiment.java** that does the following:

For the purposes of this particular question, please assume that all strings are in lowercase. You do not have to worry about any uppercase characters.

- Create an instance of StringBuilder that spells your name. Print this.
- Add, delete, or modify at least three characters in this StringBuilder object to make it spell a different word of your choice. Print this word.
- We want to append the reversed version of the original instance of the StringBuilder to

itself. Please write code to do this.

- For example if the original is “apple”, we now want to create “appleelppa”.

Question 2. Write a file called **EasterSunday.java** to compute the date for Easter Sunday using an algorithm that is described here. (You’ll be using the anonymous Gregorian algorithm--not Gauss’ Easter algorithm.)

In this program, the user will input what year it is, your code will calculate the date that Easter Sunday falls on. Then that date gets printed out in the following manner “Easter Sunday for 2018 is on April 1”.

This question does have a tiny piece of code that we have not covered yet.

Please copy these two lines of code into your EasterSunday.java file

```
String monthAsString = “April”;
if (month == 3) monthAsString = “March”;
```

These lines of code are basically saying that if it is 3rd month then write that out as March else Easter will be in April.

You will have to figure out how to use the variable monthAsString.

Note that this procedure uses a number of single character variable names. While this is not the recommended style, we will use the same variable names in this particular instance because it corresponds to a historical algorithm that would actually be harder to read if we used new variable names.

Question 3. This question asks you to design a class according to a given specification:

Design a class called Dog in a file called **Dog.java** that has the following instance variables.

Please type this code out exactly as presented here. Instead of providing starter code, we would like you to type the code yourself. Typing it out will make you think a lit bit harder about what each line is doing.

```
public class Dog {
   String name;
```

```
    String breed;
    int age;
    double weight;
}
```

Now

add the following methods to this class. We have written the constructor for you. For the rest, we are providing you with the first line. You have to fill in the rest.

- Add constructor that looks like this:

```
public Dog(String dogName, String dogBreed){
    name = dogName;
    breed = dogBreed;
    //add 2 lines here as per the instructions
}
```

The default weight of a dog is 125 g. The age of a dog upon creation (calling the constructor) should be set to 0.

Add two lines of code to the constructor above to set those defaults.

Then add these methods right below your constructor. Please ensure that you copy the first lines of these methods (the line with the method name) exactly, otherwise an automated test might fail.

```
public String getBreed() {
        // this method returns the breed of the dog
    }

public String getName() {
        // this method returns the name of the dog
    }

public int getAge() {
        //  this method returns the age
    }
```

`public double getWeight(){`
`}` – this method returns the weight
`public void eat()` {
this method increases the weight of the dog by 0.1 g
}
`public void rename(String newName)`{
this method changes the name of the dog to the new name that is supplied as an argument
}
`public void hasBirthday(){`
this method does two things:

1. It prints out ”happy birthday”

(please print exactly these 2 words. Do not change anything to uppercase.)

2. It increments the age of the dog by 1

}

What to Submit

Three files. Please make sure they are named exactly as shown below.

1. StringBuilderExperiment.java
2. EasterSunday.java
3. Dog.java - The Dog.java file does not need a main method. Students can use the main method for their own testing but the tests in Codio do not check for the existence of main.)